

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,925,638 B1
APPLICATION NO. : 09/667430
DATED : August 2, 2005
INVENTOR(S) : Koved et al.

Page 1 of 7

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 1:

Claim 1 should read as follows:

1. A method of detecting mutability of variables, objects, fields, and classes in a program component executing on a computing device including a processor and memory, said component being created in an object-oriented programming language, the method comprising the steps of:

determining whether any variable in the program component could undergo a state modification of a first type, said first type state modification being made by at least one method that is within the program component; and

performing encapsulation analysis to determine whether any variable in the program component could undergo a state modification of a second type, said second type state modification being made by at least one method that is not within the program component, to identify an exposure of the variables of the program component to modification external to the program component,

wherein a variable is mutable if its state ever changes after said variable is initialized, the state of said variable being its value together with a state of any referenced object,

an object is mutable if its state ever changes after said object is initialized, said state of said object being a set of states of all associated variables,

a field is mutable if any variable corresponding to said field is mutable, and

a class is mutable if any instance fields implemented by said class are mutable.

Column 22:

Claim 7 should read as follows:

7. A method of detecting mutability of classes in a program component executing on a computing device including a processor and memory, said component being created in an object-oriented programming language, the method comprising the steps of:

obtaining a set of classes, each of said classes being classified as one of mutable, immutable, and undecided;

testing each undecided class, said test being comprised of the sub-steps of:

testing each field in said undecided class being tested, said test being comprised of the sub-sub-steps of:

determining whether any variable corresponding to said each field could undergo a state modification of first type, said first type state modification being made by at least one method that is within said component; and

performing encapsulation analysis to determine whether any variable corresponding to said each field could undergo a state modification of a second type, said second type state modification being made by at least one method that is not within said component;

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,925,638 B1
APPLICATION NO. : 09/667430
DATED : August 2, 2005
INVENTOR(S) : Koved et al.

Page 2 of 7

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

classifying said each field as immutable if no possible first type or second type state modifications are found;
classifying said each field as undecided if there is insufficient class mutability information; and
classifying said each field as mutable otherwise;
re-classifying said undecided class as mutable if any fields in said undecided class are mutable;
re-classifying said undecided class as immutable if all fields in said undecided class are immutable;
repeating said testing each undecided class step until a number of undecided classes after a repetition of said testing step is identical to a number of undecided classes before the repetition of said testing step; and
re-classifying remaining undecided classes as mutable classes, to identify an exposure of variables of the program component to modification external to the program component.

Column:

Claim 8 should read as follows:

8. A method of detecting mutability of classes in a program component executing on a computing device including a processor and memory, said component being created in an object-oriented programming language, the method comprising the steps of:
obtaining a set of classes, each of said classes being classified as one of mutable, immutable, and undecided; testing each undecided class, said test being comprised of the sub-steps of:
testing each instance field in said undecided class being tested, said test being comprised of the sub-sub-steps of:
determining whether any variable corresponding to said each instance field could undergo a state modification of first type, said first type state modification being made by at least one method that is within said component; and
performing encapsulation analysis to determine whether any variable corresponding to said each instance field could undergo a state modification of a second type, said second type state modification being made by at least one method that is not within said component;
classifying said each instance field as immutable if no possible first type or second type state modifications are found;
classifying said each instance field as undecided if there is insufficient class mutability information; and
classifying said each instance field as mutable otherwise;
re-classifying said undecided class as mutable if any instance fields in said undecided class are mutable;

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,925,638 B1
APPLICATION NO. : 09/667430
DATED : August 2, 2005
INVENTOR(S) : Koved et al.

Page 3 of 7

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

re-classifying said undecided class as immutable if all instance fields in said undecided class are immutable;

repeating said testing each undecided class step until a number of undecided classes after a repetition of said testing step is identical to a number of undecided classes before the repetition of said testing step; and

re-classifying remaining undecided classes as mutable classes to identify an exposure of variables of the program component to modification external to the program component.

Claim 19 should read as follows:

19. A method of detecting mutability of classes and class variables in a program component executing on a computing device including a processor and memory, said component being created in an object-oriented programming, the method comprising the steps of:

obtaining a set of classes, each of said classes being classified as one of mutable, immutable, and undecided;

testing each undecided class said test being comprised of the sub-steps of:

testing mutability of each instance field in said undecided class being tested;

classifying an instance field as immutable if no possible state or encapsulation analysis modifications are found;

classifying an instance field as undecided if there is insufficient class mutability information; and

classifying an instance field as mutable otherwise;

re-classifying an undecided class as mutable if any instance fields in said undecided class are mutable;

re-classifying said undecided class as immutable if all instance fields in said undecided class are immutable;

repeating said testing each undecided class step until a number of undecided classes after a repetition of said testing step is identical to a number of undecided classes before the repetition of said testing step;

re-classifying remaining undecided classes as mutable classes; and

testing mutability of each class field in each class to identify an exposure of variables of the program component to modification external to the program component.

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,925,638 B1
APPLICATION NO. : 09/667430
DATED : August 2, 2005
INVENTOR(S) : Koved et al.

Page 4 of 7

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Claim 26 should read as follows:

26. A device for detecting mutability of variables, objects, fields, and classes in a program component executing on a computing device Including a processor and memory, said component being created in an object-oriented programming language, the device comprising: memory for holding
a layer of at least one core library and at least one data-flow analysis engine, for providing a particular abstraction of the program component;
a layer of at least one utility module, for using the results of the at least one data analysis engine to generate basic results; and
a layer of at least one mutability sub-analysis module for generating final results,
wherein a variable is mutable if its state ever changes after said variable is initialized, the state of a variable being its value together with a state of any referenced object,
an object is mutable if its state ever changes after said object is initialized, the state of an object being a set of states of all associated variables,
a field is mutable if any variable corresponding to said field is mutable, and
a class is mutable if any instance fields implemented by said class are mutable
to identify an exposure of the variables of the program component to modification external to the program component.

Claim 38 should read as follows:

38. A computer system for detecting mutability of variables, objects, fields, and classes in a program component executing on a computing device including a processor and memory, said component being created in an object-oriented programming language, the computer system comprising:
at least one computer-readable memory including:
code that determines whether any variable in the program component could undergo a state modification of a first type, said first type state modification being made by at least one method that is within the program component; and
code that performs encapsulation analysis to determine whether any variable in the program component could undergo a state modification of a second type, said second type state modification being made by at least one method that is not within the program component to identify an exposure of the variables of the program component to modification external to the program component,
wherein a variable is mutable if its state ever changes after said variable is initialized, the state of said variable being its value together with a state of any referenced object,

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,925,638 B1
APPLICATION NO. : 09/667430
DATED : August 2, 2005
INVENTOR(S) : Koved et al.

Page 5 of 7

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

an object is mutable if its state ever changes after said object is initialized, the state of said object being a set of states of all associated variables,
a field is mutable if any variable corresponding to said field is mutable, and
a class is mutable if any instance fields implemented by said class are mutable.

Claim 45 should read as follows:

45. A computer system for detecting mutability of variables, objects, fields, and classes in a program component executing on a computing device including a processor and memory, said component being written in an object-oriented programming language, the computer system comprising:

at least one computer-readable memory including:

code that obtains a set of classes, each of said classes being classified as one of mutable, immutable, and undecided;

code that tests each undecided class, said test being comprised of:

code that tests each instance field in said undecided class being tested, said instance field testing code being comprised of:

code that determines whether any variable corresponding to said each instance field could undergo a state modification of a first type, said first type state modification being made by at least one method that is within the program component; and

code that performs encapsulation analysis to determine whether any variable corresponding to said each instance field could undergo a state modification of a second type, said second type state modification being made by at least one method that is not within the program component;

code that classifies said each instance field as immutable if no possible state modifications or breakages of encapsulation are found;

code that classifies said each instance field as mutable if possible state modifications or breakages of encapsulation are found; and

code that classifies said each instance field as undecided if there is insufficient class mutability information;

code that re-classifies said undecided class as mutable if any instance fields in said undecided class are mutable;

code that re-classifies said undecided class as immutable if all instance fields in said undecided class are immutable;

code that repeats said testing each undecided class code until a number of undecided classes after a repetition of said testing code is identical to a number of undecided classes before the repetition of said testing code; and

code that re-classifies remaining undecided classes as mutable classes to identify an exposure of the variables of the program component to modification external to the program component.

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,925,638 B1
APPLICATION NO. : 09/667430
DATED : August 2, 2005
INVENTOR(S) : Koved et al.

Page 6 of 7

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Claim 53 should read as follows:

53. A computer system for detecting mutability of classes and class variables in a program component executing on a computing device including a processor and memory, said component being created in an object-oriented programming language, the computer system comprising:
at least one computer-readable memory including:
 code that obtains a set of classes, each of said classes being classified as one of mutable, immutable, and undecided;
 code that tests each undecided class, said test being comprised of:
 code that tests each instance field in said undecided class being tested, said instance field testing code being comprised of:
 code that determines whether any variable corresponding to said each instance field could undergo a state modification of a first type, said first type state modification being made by at least one method that is within the program component; and
 code that performs encapsulation analysis to determine whether any variable corresponding to said each instance field could undergo a state modification of a second type, said second type state modification being made by at least one method that is not within the program component;
 code that classifies said each instance field as immutable if no possible state modifications or breakages of encapsulation are found;
 code that classifies said each instance field as mutable if possible state modifications or breakages of encapsulation are found; and
 code that classifies said each instance field as undecided if there is insufficient class mutability information;
 code that re-classifies said undecided class as mutable if any instance fields in said undecided class are mutable;
 code that re-classifies said undecided class as immutable if all instance fields in said undecided class are immutable;
 code that repeats said testing each undecided class code until a number of undecided classes after a repetition of said testing code is identical to a number of undecided classes before the repetition of said testing code;
 code that re-classifies remaining undecided classes as mutable classes; and
 code that tests mutability of each class field in each class to identify an exposure of the variables of the program component to modification external to the program component.

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,925,638 B1
APPLICATION NO. : 09/667430
DATED : August 2, 2005
INVENTOR(S) : Koved et al.

Page 7 of 7

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Claim 60 should read as follows:

60. A computer system for detecting mutability of variables, objects, fields, and classes in a program component executing on a computing device including a processor and memory, said component being created in an object-oriented programming language, the computer system comprising:

at least one computer-readable memory including:

code that maintains a layer of at least one core library and at least one data-flow analysis engine in a mutability analyzer, for providing a particular abstraction of the program component;

code that maintains a layer of at least one utility module in a mutability analyzer, for using the results of the at least one data analysis engine to generate basic results; and

code that maintains a layer of at least one mutability sub-analysis module in a mutability analyzer, for generating final results,

wherein a variable is mutable if its state ever changes after said variable is initialized, the state of said variable being its value together with a state of any referenced object,

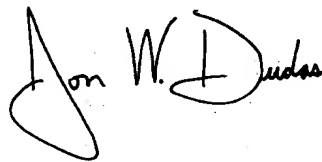
an object is mutable if its state ever changes after said object is initialized, the state of said object being a set of states of all associated variables,

a field is mutable if any variable corresponding to said field is mutable, and

a class is mutable if any instance fields implemented by said class are mutable to identify an exposure of the variables of the program component to modification external to the program component.

Signed and Sealed this

Twenty-first Day of November, 2006



JON W. DUDAS

Director of the United States Patent and Trademark Office